JOIN IN SQL

OBJECTIVES:

* To know about the SQL joins such as inner join, natural join, outer join.

OVERVIEW:

SQL JOIN clause is used to query and access data from multiple tables by establishing logical relationships between them. It can access data from multiple tables simultaneously using common key values shared across different tables.

We can use SQL JOIN with multiple tables. It can also be paired with other clauses, the most popular use will be using JOIN with WHERE clause to filter data retrieval.

Types of SQL joins:

1. SQL INNER JOIN:

The INNER JOIN keyword selects all rows from both the tables as long as the condition is satisfied. This keyword will create the result-set by combining all rows from both the tables where the condition satisfies i.e value of the common field will be the same.

Syntax:

The syntax for SQL INNER JOIN is:

SELECT table1.column1,table1.column2,table2.column1,....

FROM table1

INNER JOIN table2

ON table1.matching\_column = table2.matching\_column;

Here,

table1: First table.

table2: Second table

matching\_column: Column common to both the tables.

1. SQL Natural join:

Natural join can join tables based on the common columns in the tables being joined. A natural join returns all rows by matching values in common columns having same name and data type of columns and that column should be present in both tables. Both table must have at least one common column with same column name and same data type. The two table are joined using Cross join. DBMS will look for a common column with same name and data type Tuples having exactly same values in common columns are kept in result.

Syntax:

The syntax for SQL NATURAL JOIN is:

SELECT table1.column1,table1.column2,table2.column1,....

FROM table1

NATURAL JOIN table2

1. LEFT JOIN

LEFT JOIN returns all the rows of the table on the left side of the join and matches rows for the table on the right side of the join. For the rows for which there is no matching row on the right side, the result-set will contain null. LEFT JOIN is also known as LEFT OUTER JOIN.

Syntax

The syntax of LEFT JOIN in SQL is:

SELECT table1.column1,table1.column2,table2.column1,....

FROM table1

LEFT JOIN table2

ON table1.matching\_column = table2.matching\_column;

1. SQL RIGHT JOIN

RIGHT JOIN returns all the rows of the table on the right side of the join and matching rows for the table on the left side of the join.It is very similar to LEFT JOIN For the rows for which there is no matching row on the left side, the result-set will contain null. RIGHT JOIN is also known as RIGHT OUTER JOIN.

Syntax:

The syntax of RIGHT JOIN in SQL is:

SELECT table1.column1,table1.column2,table2.column1,....

FROM table1

RIGHT JOIN table2

ON table1.matching\_column = table2.matching\_column;

1. SQL FULL JOIN

FULL JOIN creates the result-set by combining results of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both tables. For the rows for which there is no matching, the result-set will contain NULL values.

Syntax

The syntax of SQL FULL JOIN is:

SELECT table1.column1,table1.column2,table2.column1,....

FROM table1

FULL JOIN table2

ON table1.matching\_column = table2.matching\_column;

LAB WORK

1. Create two tables Departments and Employees .

Query:

To Create Tables:

For Departments:

CREATE TABLE Departments(

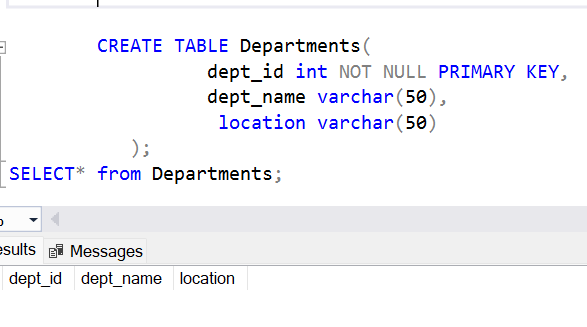
dept\_id int NOT NULL PRIMARY KEY,

dept\_name varchar(50),

location varchar(50)

);

OUTPUT



For Employees:

CREATE TABLE Employees(

emp\_id int NOT NULL PRIMARY KEY,

emp\_name varchar(50),

dept\_id int,

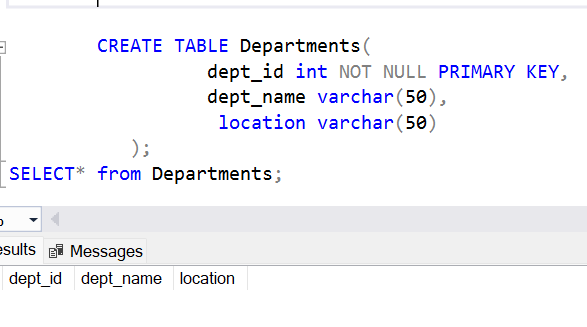
age int,

salary int

FOREIGN KEY (dept\_id) REFERENCES Departments(dept\_id),

);

OUTPUT



1. Insert the values into the tables.

INSERT INTO Departments VALUES

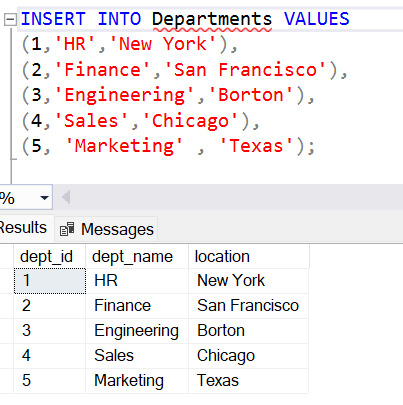
(1,'HR','New York'),

(2,'Finance','San Francisco'),

(3,'Engineering','Borton'),

(4,'Sales','Chicago'),

(5, ‘Marketing’ , ‘Texas);



INSERT INTO Employees VALUES

(1,'Anchal',1,28,60000),

(2,'Nisha',2,32,75000),

(3,'Ayush',3,26,80000),

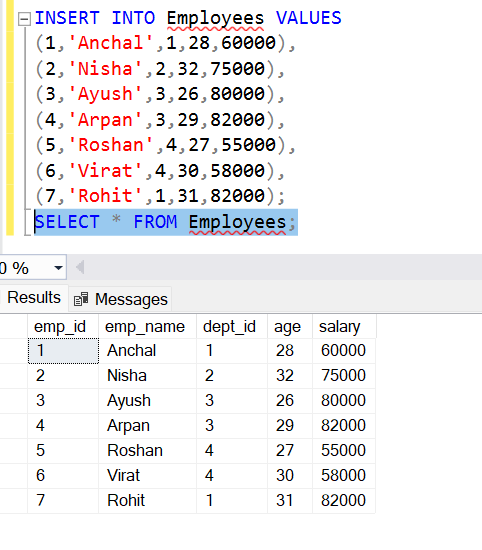
(4,'Arpan',3,29,82000),

(5,'Roshan',4,27,55000),

(6,'Virat',4,30,58000),

(7,'Rohit',1,31,82000);

OUTPUT



1. Write a Query to retrieve a list of employees along with their departments name.

Query:

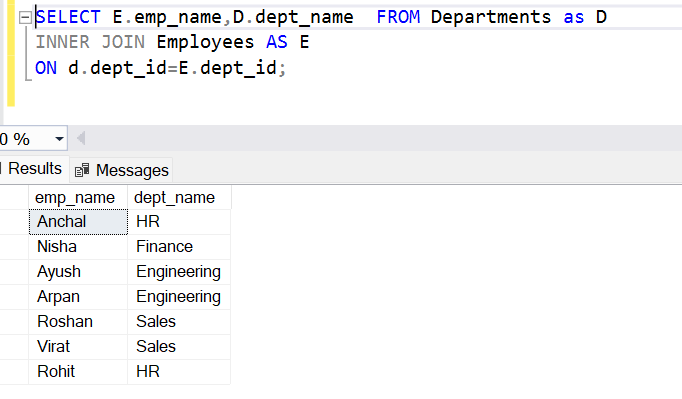
SELECT\*FROM Employees;

SELECT E.emp\_name,D.dept\_name FROM Departments as D

INNER JOIN Employees AS E

ON d.dept\_id=E.dept\_id;

OUTPUT



1. Write a Query to retrieve a list of employees and their departments

Showing employees even if not assigned to any departments .

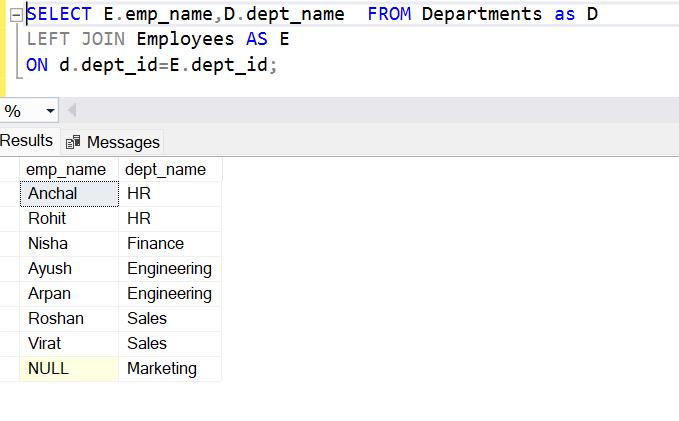
Query:

SELECT E.emp\_name,D.dept\_name FROM Departments as D

LEFT JOIN Employees AS E

ON d.dept\_id=E.dept\_id;

OUTPUT

e. Retrieve a list of all departments and their employees include departments even if they have no employees.

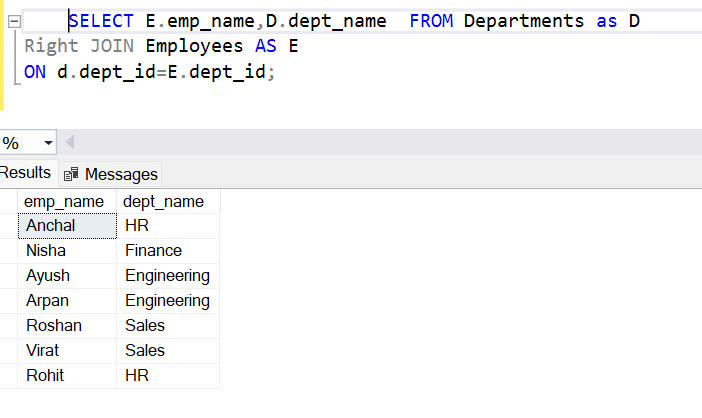
Query:

SELECT E.emp\_name,D.dept\_name FROM Departments as D

Right JOIN Employees AS E

ON d.dept\_id=E.dept\_id;

OUTPUT



1. Count the number of employees in each department.

Query:

SELECT

D.dept\_name,

COUNT(Employees.emp\_id) AS Emp\_count

FROM

Departments AS D

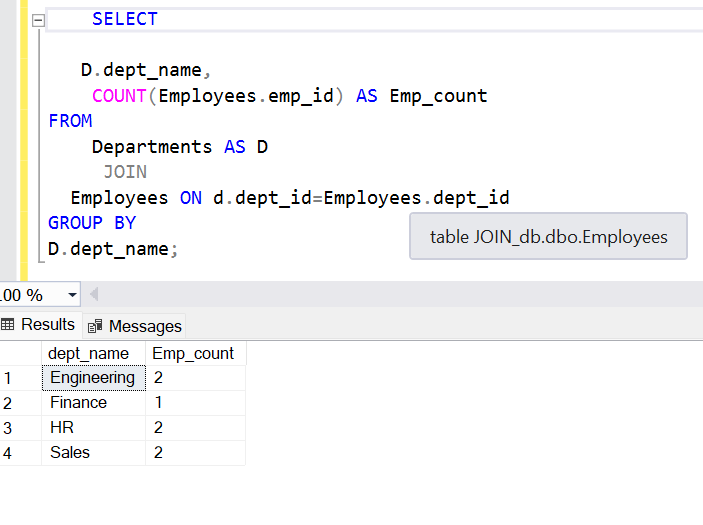
JOIN

Employees ON d.dept\_id=Employees.dept\_id

GROUP BY

D.dept\_name;

OUTPUT



1. Find the name of employees and their departments where the salary id greater than 60000.

Query:

SELECT

e. SELECT

Employees.emp\_name,

Departments.dept\_name,

Employees.salary

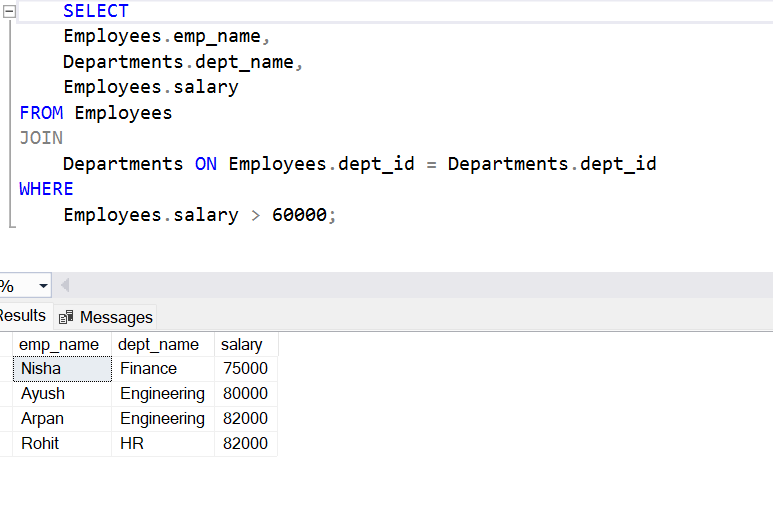
FROM Employees

JOIN

Departments ON Employees.dept\_id = Departments.dept\_id

WHERE

Employees.salary > 60000;OUTPUT



1. Find the highest paid employee in each department.

Query:

SELECT

SELECT

e.emp\_name,

d.dept\_name,

e.salary

FROM

Employees e

JOIN

Departments d ON e.dept\_id = d.dept\_id

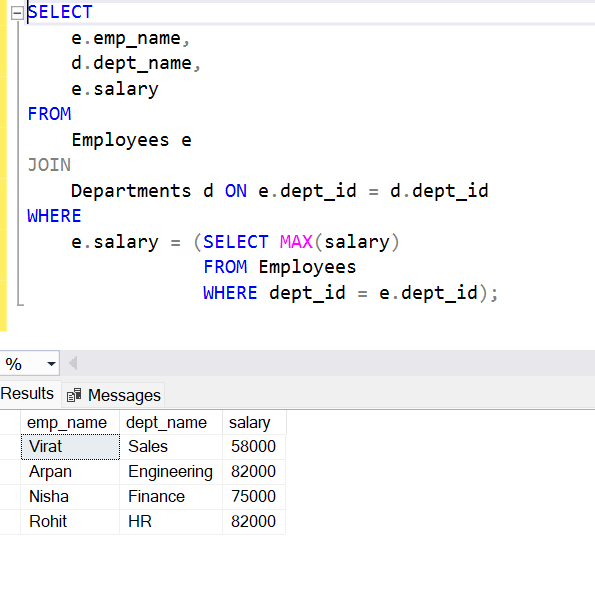
WHERE

e.salary = (SELECT MAX(salary)

FROM Employees

WHERE dept\_id = e.dept\_id);

OUTPUT



**CONCLUSION**

In conclusion, understanding SQL joins is crucial for efficient database management, enabling the combination of data from multiple tables based on related columns. Mastery of inner, outer, left, and right joins enhances data querying capabilities, providing comprehensive insights. Proper use of joins optimizes data retrieval and supports robust, scalable database applications.